Amendments to the Specification

Please replace the paragraph spanning from page 28, line 6, to page 28, line 11, (i.e., the Abstract of the Disclosure) with the following amended paragraph:

A method and system ~~access~~ for accessing multiple types of content from a broad range of client program modules is provided. A client program module can access multiple types of content without the client program module having a knowledge of what type of content it is accessing. A user can dynamically add or change services to the system. The ~~present invention~~ method and system can also serve as a standard way of exchanging content between services and client program modules.

Please replace the paragraph spanning from page 2, line 14, to page 2, line 20, with the following amended paragraph:

The present invention solves the above problems by providing a method and system for accessing multiple types of electronic content from a broad range of client program modules. A client program module can access multiple types of content without the client program module having a knowledge of what type of content it is accessing. This invention can have the flexibility to allow anything from a dictionary entry to a video to be accessed by any client program module, such as ~~Microsoft~~ Microsoft® Word and Lotus ~~Notes~~ Notes®.

Please replace the paragraph spanning from page 3, line 28, to page 4, line 2, with the following amended paragraph:

FIG. 3 is a functional block diagram illustrating yet another exemplary embodiment which can report actions between a word processing application client program module such as ~~Microsoft~~ Microsoft® Word, a service manager, some translation dictionary service containers, a cache file, and memory storage.

Please replace the paragraph spanning from page 4, line 17, to page 5, line 2, with the following amended paragraph:

The present invention solves the above problems by providing a method and system for accessing electronic content with a broad range of client program modules. This invention can provide an architecture that allows the entire system to function with any arbitrary client program module. It can have the flexibility to allow anything from a dictionary entry to a video to be accessed by any client program module, such as ~~Microsoft~~ Microsoft® Word and Lotus ~~Notes~~ Notes®. Thus, a user can access multiple types of content without the client program module having a knowledge of what type of content it is accessing because the client program module doesn't need to understand the content to be able to display it. Thus, a client program module can access new services to the system because the client program module doesn't need to know what kind of services will come along in the future. In addition, a service can be modified or expanded without disrupting its functionality. In addition, the present invention can provide a standard way of exchanging content between services and client program modules.

Please replace the paragraph spanning from page 13, line 8, to page 13, line 23, with the following amended paragraph:

FIG. 3 is a block diagram illustrating internal program module objects of an exemplary embodiment which can report actions between ~~Microsoft~~ Microsoft® Word 305, a core DLL 310, some translation dictionary service containers 370, a cache file 325, a first nonvolatile memory storage such as in INI file 215, and a second nonvolatile memory storage such as a registry 222. The core DLL 310 can be the interface of the translation dictionary service containers 370. It includes the service manager 320 and a wrapper layer 315. The service manager 320 can connect the translation dictionary services containers 370 to the wrapper layer 315, and in turn to the ~~Microsoft~~ Microsoft® Word 305. The wrapper layer 315 allows the service manager to be accessible to a client program module 205 such as ~~Microsoft~~ Microsoft® Word 305 using an interface supported by that client program module 205. The service manager 320 can perform service installation, service cataloging, and service access control. A service container 370 can be installed by an API or a setup program module. A service catalogue, created by the service manager 320, can list services available to ~~Microsoft~~ Microsoft® Word 305 based on ~~Microsoft~~ Microsoft® Word's 305 privileges.

Please replace the paragraph spanning from page 16, line 1, to page 16, line 14, with the following amended paragraph:

In one exemplary embodiment of the present invention, ~~Microsoft~~ Microsoft® Word 305 can be the client program module 205 that instantiates the service manager 320 and requests the list of available translation dictionaries service containers 370. The service manager 320 can find out the French Translation Dictionary 350, the German Translation Dictionary 375, and the Hebrew Translation Dictionary 380 are available to ~~Microsoft~~ Microsoft® Word 305. ~~Microsoft~~ Microsoft® Word 305 then can request the service manager 320 to instantiate the service objects 330 that form the available translation dictionary service containers 370. The stemmer 335, the look-up 340, and the "xml to rtf" 345 objects can be some of the service objects 330 contained in the French Translation Dictionary 350. The stemmer 335, the look-up 340, and the "xml to rtf" 345 objects can be some of the service objects 330 contained in the German Translation Dictionary 375. And the "xml to rtf" object 345 can be one of the service objects 330 contained in the Hebrew Translation Dictionary 380.

Please replace the paragraph spanning from page 16, line 17, to page 16, line 26, with the following amended paragraph:

FIG. 5 is a flow diagram illustrating an exemplary routine 405 for registering a service container 270 by an API method as set forth in FIG. 4. Registering a service container 270 can entail installing the service container 270 as part of the operating system 36. In an exemplary embodiment, the user can install ~~Microsoft~~ Microsoft® Word 305 and select the translation dictionary service container 370 as something that the user wants in ~~Microsoft~~ Microsoft® Word 305. The translation dictionary service container 370 can be installed as part of installing ~~Microsoft~~ Microsoft® Word 305 and the installation can populate the second memory storage (or registry) 222 with information about the translation dictionary service container 370.

Please replace the paragraph spanning from page 20, line 7, to page 20, line 28, with the following amended paragraph:

In an exemplary embodiment, the French translation dictionary 350 can have the following "object-chaining" order: the stemmer object 335, the look-up object 340, and the "xml to rtf" object 345. A "master-slave" relationship can exist between the stemmer object 335 and the look-up object 340, where the stemmer object 335 is a "slave" to the "master" look-up object 340. The service manager 320 can load information into a table that indicates that the stemmer object 335 is run before the lookup object 340, and the lookup object 340 is run before the "xml to rtf" object 345. The service manager 320 also can load information that indicates that the look-up object 340 is the "master" to the "slave" stemmer object 335. ~~Microsoft~~ Microsoft® Word 305 then can request the service manager 320 to instantiate each object of the desired translation dictionary service container 370 in the list. The service manager 320 then can check to see if a "master-slave" relationship exists for any of the service objects in these translation dictionary service containers 370. If yes, the service manager 320 can record the relationship. Because there is a "master-slave" relationship with the stemmer 335 and look-up 340 objects, the service manager 320 can record this. The service manager 320 then can move onto the first service 330 object in the list of chaining and can instantiate this service object 330. In this case, this can be the stemmer object 335, which has the look-up object 340 as its "master". The service manager thus can instantiate the "slave" stemmer object 335 before the "master" look-up object 340. The service manager 320 then can repeat the above steps until it reaches the last service object 330 in the list.

Please replace the paragraph spanning from page 11, line 15, to page 12, line 4, with the following amended paragraph:

Upon selection of an available service container 270, the ~~servicemanager~~ service manager 211 can construct the service container 270. The service manager 211 can connect the client program module 205 to the appropriate service container 270 and in turn the appropriate service objects 225. The service containers 270 can contain a code object 255, an associated data object 260, and a loader ID 265. The code object 255 can consist of service objects 225 that perform specific functions and that provide support to the client program module 205. Service containers 270 can be temporary arrangement of reusable service objects 225. The service objects 225 can be used by multiple service containers 270, can interact with one another, and can have predefined relationships relative to one another. In an exemplary aspect of the present invention, the service objects 225 can be linked to one another in an "object-chaining" relationship where the service objects 225 must be run in a particular order. In addition, each service object 225 can have a more dependent relationship with another service object 225 within the chain. This is a "master-slave" relationship where one service object 225 is dependent on the output of another service object 225. During execution of an object chain, a "master" object can pause the output of a service object 225 by continuously checking and accessing the output of a "slave" object until the output of the "slave" object reaches a desired threshold or value.